User based ranking algorithm for web-based distributed version control system

Jilsa Chandarana

Abstract— The field of computer science is emerging which fundamentally increases the need of programmers. One of the essential tool for a programmer is version control system. The web-bases services such as GitHub provide excellent helping hand to them. A common feature of such applications is search option. When performing a search, the items can be better sorted taking user features into account which lead to an idea of ranking algorithm specially designed for such platforms. It takes the general structure of database and convert it into graph database to easily define relation between different entities. The paper suggests a simple method for such recommendation and provides a naïve implementation in Cypher query language using Neo4j platform.

Index Terms— Algorithm, Cypher, Neo4j, Query, Ranking algorithm, User-based algorithm, VCS

1 Introduction

THE paper describes a method to score the searched items according to their relationship with user. Common to most of the approaches, the graph data structure is used and linear arithmetic operations are performed to decide the rank. The motivation for developing this idea came from personal experience while searching for a repository on web-based version control system. Upon executing the search query, tons of results were presented and required solution was later in the list which took time and efforts to find. The idea was originated from there and a problem statement was defined as constructing a ranking algorithm for sorting the results of version control system according to user preference so user will have high chances to get required item on the top.

2 BACKGROUND

2.1 Version Control Systems

A version control system is a very useful tool for software development and it is widely adopted in the industry now. It helps the developer to keep track of continuous changes occurring in a project. It has many applications in the field of software merging, collaboration modelling, software changes, software branching, open-source software projects, curriculum development etc.[5] It gives the developers an upper hand in keeping a backup and maintaining the modifications. There are two types of version control systems available. 1. Centralized Version Control System (CVCS) and 2. Distributed Version Control System (DVCS).

- Centralized Version Control System:
    - In centralized version control system, the repository is stored at single place like a server and every user can access that using their local system. Every time user needs to commit the changes, they need to contact the server. The popular examples of CVCS are Concurrent Version System (CVS), perforce and subversion.
- Distributed Version Control System:
    - On contrary, the distributed version control system copies the entire codebase to every developers' local system. Its major advantage is that users can work offline which is way faster. It is the backbone of Open Source Software (OSS). Some famous examples of DVCS are Git and Mercurial.[1] With the evolution of distributed version control system, some DevOps software emerged such as GitHub and GitLab. They are web-based platforms which help users with code management. Basic information about such platforms is given below.

• *Jilsa is currently pursuing bachelors degree program in Computer Science in Charotar University of Science & Technology, India, PH- 02697 265 011. E-mail: 18dcs010@charusat.edu.in*

2.2 GitHub

GitHub is a platform for internet hosting and software development. For underlying version control, it uses Git. Its development began in 2007 in Ruby on Rails and it was launched in 2008. It was taken over by Microsoft in 2018. It is commonly used for open source projects and collaborating with them. The user can easily upload their code to GitHub and other developers can collaborate them. Moreover, the static websites are also easily hosted on this.[2]

2.3 GitLab

GitLab was an open-source project released under MIT license. It is a cloud native application which makes it highly secure. Initially it was developed using Ruby but as time passes, some of its features were also created using GO. It allows the user to check project development using charts. It puts the restriction on number of private repository a user can have along with integrated API and third party server. It provides useful tool for entire development team or DevOps for efficient workflow. The popular users of GitLab are IBM, NVIDIA, GNOME, Goldman Sachs etc.[3]

2.4 BitBucket:

BitBucket is another competitor managed by Atlassian. It also works on Git version control and provides free accounts as well as commercial plans for unlimited private repositories. Unlike GitHub and GitLab which are written in Ruby, BitBucket is written in Python and Django framework. It was released in 2012 and it was not an open source project.[4]

2.5 Ranking Algorithm

Ranking algorithms are crucial part of search engines. Once the search engine fetches all the required documents, ranking algorithms decides the order to sort the items to show as final output to the user. The result of ranking algorithm highly affects the user experience and satisfaction. Suppose you search for a website in a search engine that you've visited many times before. The search engine finds a lot of websites with similar names and without a ranking algorithm, your website may show up at last page. Over the time, many ranking algorithms have been developed and actively used in the market to provide exceptional service to a user. Some of the famous ranking algorithms have been discussed here.[5]

3 Literature Review

The history of ranking algorithm goes back to InDegree algorithm that formed the basis for its successors. It was calculating ranks based on number of links pointing to a webpage. Following that, PageRank came into the existence. PageRank is one of the most famous ranking algorithm used by Google. It works on the graph structure of internet. It takes the incoming and outgoing links of webpages into consideration to rank the webpages. If a webpage has many backlinks or it has few highly ranked backlinks pointing to it then it will get a higher score and will be shown on the top. PageRank can be computer for any size of collection of documents. It uses an iterative approach to reach to the output.[6] HITS is abbreviation of Hyperlink-Induced Topic Search. HITS algorithm gives equal importance to backlinks as well as forward links. It uses the concept of Hub and Authorities. It is said that "a good hub is a page that points to many good authorities, and a good authority is a page that is pointed to by many good hubs". It finds the subset of containing relevant hubs and authorities before calculating the score. That's why if the query changes, the resulting subset changes so it's query sensitive.[7] SALSA stands for Stochastic Approach for Link-Structure Analysis. It is an improved version of HITS which results in reduction of response time of algorithm. It is similar to PageRank in the sense that it uses random walk through Markov Chains that is a chain of hubs and a chain of authorities. It shows better immunity again Tightly Knit Community (TKC) than HITS. Twitter uses SALSA algorithm to recommend other users to follow.[8] In recent time, many ranking algorithms have been developed as customized solutions. As suggested by P. Ghosh et al.[9] the PageRank algorithm can be modified by taking location and time into consideration. The terms were simply added together to calculate the rank. They stated that number of hitting of pages also plays an important role. In 2020, L. J. Sankpal et al.[10] presented a re-ranking algorithm which used links collected from different web browsers and after preprocessing, the feature extraction was performed to arrange the pages according to fitness measures. Huda Alghamdi et al.[11] proposed a weighted algorithm which also generated the web graph by the data provided by crawler. In this method, each page is assigned an initial rank and some mathematical operations were performed to decide its final rank. Again, the situation decides which platform is best for your use but there are some common features that are taken into the account for generalization of this method. In web-based applications like there, users can create their account to keep track of their section. They can follow each other for further details and create their own repositories. Repositories are nothing but the project directory which contains all the required files. The repositories can be forked and pull requests can be sent. The grouping option is also provided to work with limited and chosen developers. Most of the ranking algorithms work on graphical structure of web. Generally speaking, the structure of web-based version control system database can be converted to a graph. The nodes can be of various types such as user, repositories or organization. That forms the bottom line of this paper. Such platforms store millions of repositories of millions of users. The search option is provided to find the required repository or user quickly. But due to plenty of options, it may get inconvenient to find required item from list of thousand. That's why this paper suggests a method to list items based on user preferences. Out of all the matching repositories or users, the algorithm will score them based on user-preference and decide its ranking.

4 Proposed Approach

Before using this searching approach, it is assumed that the
required items were already filtered. Like by searching a keyword, list of all matching repositories or users has been
found i.e. indexing has been performed by any possible methods.[12] Applying this algorithm after filtering reduces
the processing time.
The algorithm is based on common factors of version control
system including user follow, collaborators, organizations
starred repositories, forked repositories and pull requests. Here users and repositories are nodes of the graph so it is assumed that username of a user will be unique.

5 Algorithm

Input: The data of existing users, repositories and organization

Output: A list of searched items sorted by user preference

1. Create the graph of existing organizations, users and repositories with their corresponding attributes like organization name and established year for organization... username, organization, contact info etc. for users and title, owner, total stars, id (username/title for uniquely identify a repository) etc. for repository.

2. Join the nodes with various types of edges. Like an edge from a user who follows another user, an edge from a

repository to parent repository which was forked, an edge from a user to repository which they starred etc.

Step 2 will produce a directed graph.

3. The static weights are assigned as requirement to all the edges.

4. The user which is searching is the starting node. The score for all repositories or users is computed as follow.

$$\frac{\sum_{e \in path} \frac{weight(e)}{index(e)}}{Total\ number\ of\ paths}$$

With increase in distance, the relevance decreases, so the weight of an edge is divided by its index i.e. the position it appears in the path so that the further edges don't influence much in the result.

As the number of paths increases, the sum will increase too. That's why it is then divided by number of paths to take average of them.

5. Sort the items from score as an output.

6 Naïve Implementation and Results

In this implementation, a dummy dataset is used. The CSV files of data is uploaded to google drive and loaded into Neo4j for further processing. Three types of nodes are created named Organization, User and Repository. The loading of data is done using LOAD CSV as specified in below query.

```
// Loading organizations

LOAD CSV WITH HEADERS FROM
"https://drive.google.com/uc?export=download&id=1PKEp0
AXvXM4p-VOL-c8MSoOHLhJNG7ou" AS row

CREATE (org: Organization {name: row.Organizations,
established: row.Established})


// Loading Users who do not belong to any organization

LOAD CSV WITH HEADERS FROM
"https://drive.google.com/uc?export=download&id=1ul8ichc
X3xyIgG3B3fF1cF0t4wpp8wOf" AS row

WITH row WHERE trim(row.organization) IS null

CREATE (usr: User {name: row.name})
```

```
// Loading users who belong to some organization and they
are connected with corresponding organization node

LOAD CSV WITH HEADERS FROM
"https://drive.google.com/uc?export=download&id=1ul8ichc
X3xyIgG3B3fF1cF0t4wpp8wOf" AS row

WITH row WHERE trim(row.organization) IS NOT null

MATCH (org: Organization {name: row.organization})

MERGE (usr: User {name: row.name})

CREATE (usr)-[:organization]->(org)
```

There are six types of relations are created. Namely "createdBy" to relate repository to their creator, "follow" to show which user follows whom, "collaborate" to check which user collaborates to which repository, "starredBy" which indicates which users have starred any repository, "forked" to connect child repository to its parent and "pullrequest" which points to repositories which sent pullreqquest to other repositories.
They are also noted in CSV file and loaded in the same way.

```
// Loading "createdBy" relation
LOAD CSV WITH HEADERS FROM
"https://drive.google.com/uc?export=download&id=1F7e_t
Me6cWUAI-mzgKLQymMXa61wBLfg" AS row
MERGE (usr: User {name: row.user})
CREATE (repo: Repository {title: row.title, owner: row.user,
keywords: split(row.keywords,","), id: row.id})-[:createdBy]-
>(usr)

// Loading "follow" relation
LOAD CSV WITH HEADERS FROM
"https://drive.google.com/uc?export=download&id=1rKYM
_ZeSgspTzhtCCZofvg_l0D7sv6Nf" AS row
MERGE (usr: User {name: row.user1})
MERGE (usr2: User {name: row.user2})
CREATE (usr)-[:follow {weight: row.weight}]->(usr2)


// Loading "collaborate" relation
LOAD CSV WITH HEADERS FROM
"https://drive.google.com/uc?export=download&id=1XDXM
GUhEOzcyXok2OpIL-b18DJh3sCsF" AS row
MATCH (repo: Repository {title: row.title})-[:createdBy]-
>(usr: User {name: row.user}), (usr2: User {name:
row.user2})
CREATE (repo)-[:collaborate {weight: row.weight}]->(usr2)


// Loading "starredBy" relation
LOAD CSV WITH HEADERS FROM
```

"https://drive.google.com/uc?export=download&id=1WkWmC7i0WQWqO5JXQ47fcc1nBgp0aZ-6" AS row
MATCH (repo: Repository {title: row.title})-[:createdBy]->(usr: User {name: row.user2}), (usr2: User {name: row.user})
CREATE (repo)-[:starredBy {weight: row.weight}]->(usr2)

// Loading “forked” relation
LOAD CSV WITH HEADERS FROM "https://drive.google.com/uc?export=download&id=1Krc0mhgY-LIxUfjyrUqiIOrhBAEWqLPn" AS row
MATCH (repo: Repository {title: row.title})-[:createdBy]->(usr: User {name: row.user}), (repo2: Repository {title: row.title2})-[:createdBy]->(usr2: User {name: row.user2})
CREATE (repo)-[:forked {weight: row.weight}]->(repo2)

// Loading pullrequest” relation
LOAD CSV WITH HEADERS FROM "https://drive.google.com/uc?export=download&id=1pKcb7GXigao_zTka4URs4Tt1hQFXzygE" AS row
MATCH (repo: Repository {title: row.title})-[:createdBy]->(usr: User {name: row.user}), (repo2: Repository {title: row.title2})-[:createdBy]->(usr2: User {name: row.user2})
CREATE (repo)-[:pullrequest {weight: row.weight, status: row.status}]->(repo2)

We can also explicitly give weights to any edges.

// Assigning 5 unit weights to “organization” edge
MATCH (:User)-[org: organization]->(:Organization)
SET org.weight = 5

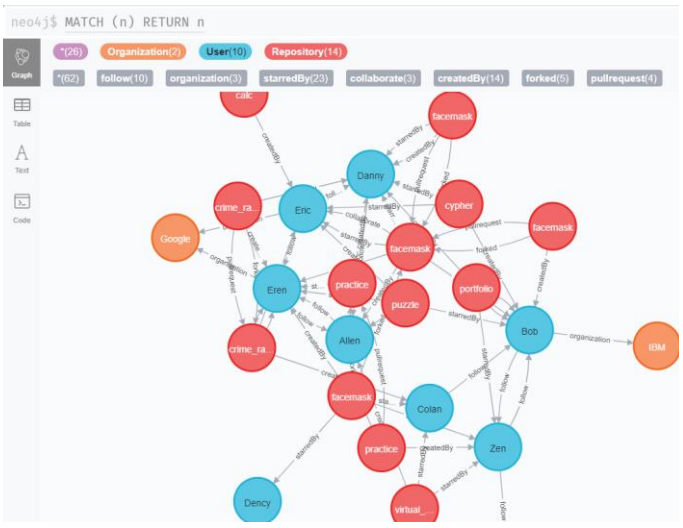
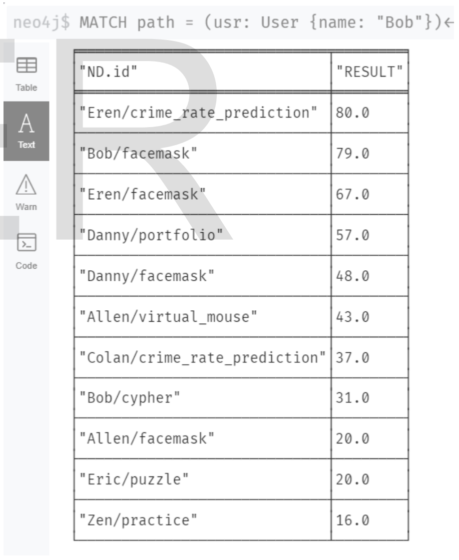


Fig 1. Graph Visualization

To make our future work easy, we can convert all the weights to integer so arithmetic operations can be performed on them without any problem.
// Converting weights to integers
MATCH ()-[r]->()
SET r.weight = toInteger(r.weight)

Now that our graph is ready, we can query our database to show us list of recommended items.
To search for repositories recommended for user “Bob”, the query can be in following form.
MATCH path = (usr: User {name: "Bob"})<-[x*]-(repo: Repository)
UNWIND relationships(path) AS rels
WITH sum(rels.weight / (apoc.coll.indexOf(relationships(path), rels) + 1)) as SCORE, last(nodes(path)) AS ND
RETURN ND.id, avg(SCORE) as RESULT
ORDER BY RESULT DESC

We are finding all paths between Bob and all repositories which will be stored in path variable. Path is the list of all possible paths so we will take one path at a time and compute weight / index of edge. Once that is computed, we will average all the score given and list them which will be returned along with ID of repository. The output of this query in text form is given in fig 2.

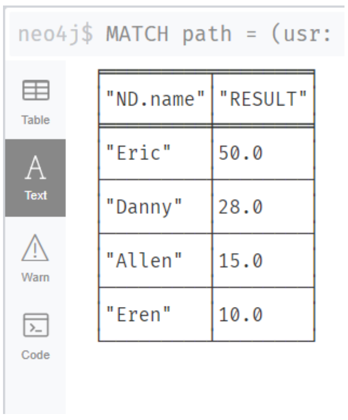


```
neo4j$ MATCH path = (usr: User {name: "Bob"})<
```

"ND.id"	"RESULT"
"Eren/crime_rate_prediction"	80.0
"Bob/facemask"	79.0
"Eren/facemask"	67.0
"Danny/portfolio"	57.0
"Danny/facemask"	48.0
"Allen/virtual_mouse"	43.0
"Colan/crime_rate_prediction"	37.0
"Bob/cypher"	31.0
"Allen/facemask"	20.0
"Eric/puzzle"	20.0
"Zen/practice"	16.0

Fig. 2. Recommended Repositories for user named “Bob”

Similarly, if user “Allen” is searching for user, the score is computed in the same way.
MATCH path = (usr: User {name: "Allen"})<-[x*]-(usr2: User)
UNWIND relationships(path) AS rels WITH sum(rels.weight / (apoc.coll.indexOf(relationships(path), rels) + 1)) as SCORE, last(nodes(path)) AS ND
RETURN ND.name, avg(SCORE) as RESULT
ORDER BY RESULT DES



```
neo4j$ MATCH path = (usr:
```

"ND.name"	"RESULT"
"Eric"	50.0
"Danny"	28.0
"Allen"	15.0
"Eren"	10.0

Fig. 3. Recommended users for user named “Bob”

As intended, the algorithm scores the items accordingly and sort them. But as shown, due to self-loops and unadjusted weights, the user himself may get less score than their closest neighbors.
The algorithm gives output that consists of connected nodes. It can be assumed that rest of nodes are at equal and least priority from user’s point of view.

7 Advantages

The main advantage of this algorithm is its simplicity. No costly operation is performed. It can be naively implemented using aggregate functions only.
Another big advantage is its flexibility. It is highly customizable as the weight parameter can be modified. Changing the weights will yield a different result. We can fine-tune the parameters in future by analyzing user requirements as well. And even in cases such as pullrequest, different weights can be assigned according to their status such as “Active” or “Close”. The self-loops can also be added to prioritize the user more.

8 Disadvantages

Finding all possible paths in huge dataset can be time and resource consuming. It may not respond in expected time in such case. To solve that problem, the maximum length of path can be fixed so that items further away from that threshold is not considered for calculation.

9 Conclusion

In this paper, a method was proposed to rank items from search option especially designed for web-based version control applications. As graph datasets are evolving, the graph database is chosen for implementation. Neo4j is leading platform for such purpose with variety of services and extra ordinary user experience. The method is highly customizable which offers the user to change its parameters as requirements. But it can provide a light weight solution as ranking algorithm especially for small to moderate graphs.

References

C. Rodríguez-Bustos and J. Aponte, "How Distributed Version Control Systems impact open source software projects," 2012 9th IEEE Working Conference on Mining Software Repositories (MSR), 2012, pp. 36-39, doi: 10.1109/MSR.2012.6224297.

[2]. F. Chatziasimidis and I. Stamelos, "Data collection and analysis of GitHub repositories and users," 2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA), 2015, pp. 1-6, doi: 10.1109/IISA.2015.7388026.

[3]. Cortés Ríos, Julio César & Kopec-Harding, Kamilla & Eraslan, Sukru & Page, Christopher & Haines, Robert & Jay, Caroline & Embury, Suzanne. (2019). A Methodology for Using GitLab for Software Engineering Learning Analytics.

[4]. Puthea, Khem & Lyta, Ly. (2017). The comparison of collaboration and communication software. 10.13140/RG.2.2.13995.44326.

[5]. Signorini, Alessio. (2005). A Survey of Ranking Algorithms.

[6]. Patel, Punit & Patel, Kanu. (2015). A Review of PageRank and HITS Algorithms. International Journal of Advance Research in Engineering, Science & Technology. 2. 2394-2444.

[7]. International Journal of Engineering Research & Technology (IJERT), Vol. 1 Issue 8, October – 2012, ISSN: 2278-0181

[8]. Lempel, R. & Moran, Shlomo. (2001). SALSA: The stochastic approach for link-structure analysis. ACM Trans. Inf. Syst.. 19. 131-. 10.1145/382979.383041.

[9]. P. Ghosh and S. Sen, "Time and location based summarized PageRank calculation of Web pages," 2014 IEEE International Conference on Industrial Technology (ICIT), 2014, pp. 788-791, doi: 10.1109/ICIT.2014.6894920.

[10]. L. J. Sankpal and S. H. Patil, "weWeb Page Re-Ranking using Squirrel Search Rank Algorithm," 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), 2020, pp. 271-278, doi: 10.1109/ICISS49785.2020.9315998.

[11]. H. Alghamdi and F. Alhaidari, "Extended User Preference Based Weighted Page Ranking Algorithm," 2021 National Computing Colleges Conference (NCCC), 2021, pp. 1-6, doi: 10.1109/NCCC49330.2021.9428844.

[12]. Kalyani, Darshita & Mehta, Dr. (2017). Paper on Searching and Indexing Using Elasticsearch. International Journal Of Engineering And Computer Science. 10.18535/ijecs/v6i6.45.